IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

RESPONSE B **TO OFFICE ACTION MAILED 04/10/2006**	**Application SC/File No.**	09/992,120
	Filing Date	11/14/2001
	First Named Inventor	Ronald Hilton
	Group Art Unit	2128
MAIL STOP: AMENDMENT Commissioner for Patents P.O. Box 1450 Alexandria, VA 22313-1450	**Examiner Name**	Saxena, Akash
	Attorney Docket No.	AMDH-08157US0 DEL
	Confirmation No.	4631
	Customer No.	21603

In response to the Office action of 04/10/2006, please amend the above-identified application as follows:

Claims are described on page 2 of this *RESPONSE B* and include a listing of all claims now in the application.

Remarks/Arguments begin on page 5 of this *RESPONSE B*.

LISTING OF CLAIMS AFTER *RESPONSE B*

Claims

This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims

1. (Previously Amended) A computer-implemented method for dynamic emulation of legacy instructions of a legacy program comprising:

providing state information for determining a program execution mode for emulating said legacy instructions,

accessing said legacy instructions and said state information,

for each particular legacy instruction,

querying to determine if one or more particular translated instructions for said execution mode are stored as a result of translating said legacy instruction for said execution mode, and

if not translated for said execution mode,

translating the particular legacy instruction into one or more particular translated instructions for emulating the particular legacy instruction for said execution mode,

storing said one or more particular translated instructions with said state information, and

if translated for said execution mode, continuing without additional translating,

accessing said one or more particular translated instructions for emulating said legacy instructions for said execution mode;

executing said translated instructions to emulate said legacy instructions.

2. (ORIGINAL) The method of Claim 1 wherein said storing of the one or more particular translated instructions is in one or more particular translated blocks and said state information is stored in each of said particular translated blocks.

3. (ORIGINAL) The method of Claim 1 wherein said legacy instructions are for a legacy system having a S/390 architecture.

4. (ORIGINAL) The method of Claim 1 wherein said legacy instructions are object code instructions compiled/assembled for a legacy architecture.

5. (ORIGINAL) The method of Claim 1 wherein said translated instructions are for execution in a RISC architecture.

LISTING OF CLAIMS AFTER *RESPONSE B*

6. (Previously Amended) A computer-implemented method for dynamic emulation of legacy instructions, where the legacy instructions are compiled/assembled into object code form for a native architecture, where the legacy instructions are executed as guests in the host architecture, where the legacy instructions are translated to translated instructions in the host architecture and the translated instructions are executed in the host architecture concurrently with the translation of the legacy instructions in the host architecture, comprising:

providing state information for determining a program execution mode for emulating said legacy instructions,

accessing said legacy instructions and said state information as guests in the host architecture,

for each particular legacy instruction,

querying to determine if one or more particular translated instructions for said execution mode are stored as a result of translating said legacy instruction for said execution mode, and

if not translated for said execution mode,

translating the particular legacy instruction into one or more particular translated instructions for emulating the particular legacy instruction for said execution mode,

storing said one or more particular translated instructions with said state information, and

if translated for said execution mode, continuing without additional translating,

accessing said one or more particular translated instructions for emulating said legacy instructions for said execution mode as a guest in said host architecture,

executing said translated instructions to emulate said legacy instructions.

Remarks/Arguments

Office Action Summary

Status.

1. This *RESPONSE B* is in answer to the Office communication mailed 04/10/2006.
2. The Office communication is non-final.
3. NA

Disposition of Claims.

4. Claims 1 - 6 are pending in the application.
5. No Claims have been allowed.
6. The rejected Claims 1 - 6 have previously been amended.
7. NA
8. NA

Application Papers.

9. NA
10. NA
11. NA

Priority under 35 U.S.C. § 119.

12. NA

DETAILED ACTION

1. Claims 1-6 as amended remain in the application
2. *RESPONSE A* filed 10th February 2006 amended Claims 1-6.
3. Claims 1-6 as amended remain rejected.

Response to Applicant's Remarks & Examiner's Withdrawals

1. The Examiner's withdrawal of the claim rejection(s) under 35 USC § 101 is noted.
2. The Examiner's withdrawal of the claim rejection(s) under 35 USC § 102 to claim(s) 1-2, 4 & 6 is noted.
3. The Examiner's withdrawal of the claim rejection(s) under 35 USC 5 103 to claim(s) 3 & 6 is noted.
4. The Examiner's further, response to claim interpretation is noted.

Response to Applicant's Remarks Claim Interpretation

5. The Examiner argues as follows:

5.1.

> *Applicant has argued that "program mode of operation" as interpreted by the examiner, is stored in the PSW and control registers in IBM S/390 architecture, is an interpretation believed to in error.*
> *Examiner would like to point out that in specification that PSW and control register store "program execution" information (Specification: [0027]). Further, program event recorder (PER) is enabled by PSW and control registers (Specification:[0030]). The PER directly influences the generation of STATE word (Specification: [0033]-[0035]).*

5.2. The Examiner's argument as quoted in Section 5.1 does not accurately portray Applicant's argument as set forth in prior *RESPONSE A*. The Examiner seems to miss the point as argued in *RESPONSE A* that information stored in the "*PSW and control registers in IBM S/390 architecture*" may be accurate at one point in time but is not accurate at a later point in time.

5.3. Applicant's argument in *RESPONSE A* appeared therein in paragraph 2.2 as follows:

> *To the extent that the Examiner's interpretation is understood, it is believed in error. As described in the Specification (Page 6-7 [0027]), the PSW and control register information changes from time to time during execution. The particular PSW information when legacy code is first translated to translated code may have first values, but when that legacy code again appears for execution a second time, the PSW information will have second values with no guarantee that the second values are the same as the first values.*

5.4. As quoted in Section 5.3, Applicant was clearly arguing that the Examiner's error was in not recognizing that information stored in the "*PSW and control registers in IBM S/390 architecture*" may be accurate at one point in time but is not accurate at a later point in time. Therefore, Applicant's conclusion in Paragraph 2.2 of *RESPONSE A* that **at the time of re-execution** of the previously translated code "…, the required information for proper execution of the translated code is nowhere stored in the PSW and control registers."

5.5. The Examiner by omission in the quotation of Section 5.1 above suggests that Applicant's are arguing that the "program mode of operation" is never stored in the "*PSW and control registers in IBM S/390 architecture*". Applicant is not making an argument that the "program mode of operation" was never stored in the "*PSW and control registers"* but is arguing that **at a subsequent time** for re-execution of translated code, there is no guarantee that the correct "program mode of operation" is stored in the "*PSW and control registers".*

Remarks/Arguments

5.6. The Examiner further in Paragraph 5 of the 04/10/2006 Office action wrongly misstates Applicant's argument as follows:

> *Further, applicant has presented contradictory arguments, as follows. In remarks (2.2 and 4.2.2) applicant has stated:*
>
> > *2.2 To the extent that the Examiner's interpretation is understood, it is believed in error. As described in the Specification (Pane 6-7 (0027j), the PSW and control resister information changes from time to time during execution. The particular PSW information when legacy code is first translated to translated code may have first values, but when that legacy code again appears for execution a second time, the PSW information will have second values with no guarantee that the second values are the same as the first values. If different, execution of the already translated code based upon the PSW information second values will lead to incorrect execution since the translated code was based on PSW information first values. Accordingly, the required information for proper execution of the translated code is nowhere stored in the PSW and control resisters.*
> > *4.2.2 From the specification, and particularly the quotes in Section 4.2. 1 above, "program execution mode" in the present application is referring to the overall architectural mode of the computer as it relates to programs being executed. These pronram execution modes are set, for example in the S/390 architecture in the Proaram Status Word (PSW), and are not decipherable by inspecting the individual instructions or any instruction state information used for individual instructions.*
>
> *This appears to be circular reasoning because the architecture under discussion is S/390 in the specification and it can not be asserted both ways that PSW does and does not store the program execution information to make argument against two separate issues using same architecture. Further, the second argument reinforces the examiner's argument for PSW storing program execution information.*

5.7. The Examiner's argument of "circular reasoning" completely misses the point that **at the time of execution** of the translated instruction, the required information is nowhere reliably stored in the "*PSW and control registers*". While the required information, of course, was stored in the system **at the time of original translation**, that information becomes lost and is not reliably present subsequently **at the time of execution** of the translated instruction. It is not circular reasoning to state that the required information is not guaranteed to be present when it is needed. The Examiner does not argue, since it is not true, that the "*PSW and control registers*" have the required information **at the time of execution** of the translated instruction.

Response to Applicant's Remarks Claim Interpretation

6. The Examiner concludes that the prior arguments are moot. Applicant agrees that they are moot, but also the Examiner's arguments are in error for the reasons presented and hence have no probative value with regard to the new rejections.

Claim Rejections - 35 USC § 102

7. The rejection of Claim 1-3 and 5 under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 5,577,231 issued to Scalzi et al is noted. This rejection is respectfully traversed for the following reasons.

7.1. Scalzi is limited to a dynamic address translation (DAT) mechanism in which Effective Addresses (EA) normally generated by the DAT are expanded to a large number of Virtual Effective Addresses (VEA) by appending State information to the Effective Address (EA) to form the VEA addresses for the "target virtual address space" of the target machine. The preferred embodiment of Scalzi uses the entire 64-bit address range of the target system for VEA's of a 32-bit source system and in every case, the VEA address space must be much larger than the VE address space. Scalzi does not store state information with translated instructions, but rather creates an entirely new VEA address location for every different change in state information.

Remarks/Arguments

7.2. By way of distinction, the present invention translates source instructions and then stores the state information with the translated instructions. The amount of additional memory is small compared to the operation of Scalzi. The present invention operates independent of whether a DAT facility is present or not whereas Scalzi is only for a DAT mechanism.

7.3. Referring to Claim 1, for example, the following recitations appear:

> *translating the particular legacy instruction into one or more particular translated instructions for emulating the particular legacy instruction for said execution mode,*
>
> *storing said one or more particular translated instructions with said state information, and*

7.4. Referring to the "translating element" in Section 7.3, the Examiner argues,

> *translating the particular legacy instruction into one or more particular translated instructions for emulating the particular legacy instruction for said execution mode* is shown by *Scalzi: Co1.6 Lines 6-59 - dynamic address translation (DAT) process checking the page table; Col.)*

7.5. However, *Scalzi: Co1.6 Lines 6-59* has no discussion whatsoever of translated instructions but only discusses addresses and DAT.

7.6. The Examiner further argues with respect to the "storing element" of Section 7.3,

> *storing said one or more particular translated instructions with said state information as storing the target address and state information associated to source code in target virtual address which is associated to translated code (Scalzi: Co1.7 Lines 26-31; Co1.8 Lines 58-Co1.10 Lines 8),*

7.7. However, *Scalzi: Co1.7 Lines 26-31; Co1.8 Lines 58-Co1.10 Lines 8* has no discussion of storing translated instructions or storing any state information with such instructions. Apparently the Examiner is somehow attempting to equate addresses and code, but no such equality is discussed or suggested in Scalzi as argued by the Examiner.

7.8. The actual operation of Scalzi is to exhaustively expand the target virtual address space treating state information as an address expander in a DAT mechanism. By

way of distinction, the present invention stores state information with one or more translated instructions without need of the extraordinary address expansion of Scalzi (and without need of a DAT facility although such DAT facility can be employed with systems using the present invention). Scalzi does not store state information, but rather creates new address locations for storing multiple synonym virtual addresses that can map to the same real address. The present invention does not generate synonym virtual addresses. The Examiner is apparently wrongfully relying on Applicants disclosure to find a correspondence between elements since no such correspondence is found in Scalzi.

7.9. Upon review, it is believed that in light of the above remarks, the Examiner will now find Claim 1-3 and 5 allowable at least for the above reasons.

Regarding Claim 2

7.10. The Examiner argues that

> *Scalzi teaches storing of the one or more particular translated instructions is in one or more particular translated blocks and said state information is stored in each of said particular translated blocks as page frames (Scalzi: Co1.6 Lines 6-30) as DAT based translation and exploded virtual target effective address (VEA) (Scalzi: Co1.9 Lines 32-62; Co1.24 Lines 55-65).*

7.10.1. Notwithstanding the Examiner's argument, the sections of Scalzi make no reference to storage of translated blocks or state information therefore. If translated instructions are stored, why do they not spread across block boundaries and how is state information allocated to blocks. In fact, no state information is stored at all in any blocks.

Regarding Claim 3

7.11. Claim 3 is believed allowable at least for the reasons of Claim 1.

Regarding Claim 5

7.12. Scalzi teaches translated instructions are for execution in a RISC architecture (Scalzi: Co1.9 Lines 13-21).

7.12.1. Claim 5 is believed allowable at least for the reasons of Claim 1.

Remarks/Arguments

Claim Rejections - 35 USC § 103

8. Claims 4 & 6 were rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,577,231 issued to Scalzi et al (Scalzi hereafter), further in view of U.S. Patent No. 6516295 issued to George A. Mann et al (Mann hereafter).

8.1. As discussed above, Scalzi is limited to a DAT implementation and otherwise does not operate. The Examiner has not discussed and it is not apparent how the Mann system would function in a DAT environment. Accordingly, the combination of the Mann and Scalzi references would not produce any operable system or at the very least would in now way suggest Applicant's invention as set forth in the claims.

8.2. Upon reconsideration it is believed the Examiner will find all claims allowable.

Respectfully submitted,

	SIGNATURE	
David E. Lovejoy (US Reg. No.: 22,748)	David E. Lovejoy **/david lovejoy/**	**Signature Date** 10 October 2006
Mail Address	Customer No.	Communication
102 Reed Ranch Rd. Tiburon, CA 94920-2025 USA	21603	Tel: (415) 435-8203 Fax: (415) 435-8857 e-mail:david.lovejoy@sbcglobal.net